Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Hydras: Complexity on general graphs and a subclass of trees

Petr Kučera

Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University in Prague, Malostranské nám. 25, 118 00 Praha 1, Czech Republic

ARTICLE INFO

Article history:
Received 27 March 2015
Received in revised form 29 February 2016
Accepted 23 May 2016
Available online xxxx

Keywords:
Horn CNF
Horn minimization
Hydra formula
Caterpillar

ABSTRACT

Hydra formulas were introduced in [1]. A hydra formula is a Horn formula consisting of definite Horn clauses of size 3 specified by a set of bodies of size 2, and containing clauses formed by these bodies and all possible heads. A hydra formula can be specified by the undirected graph formed by the bodies occurring in the formula. The minimal formula size for hydras is then called the *hydra number* of the underlying graph. In this paper we aim to answer some open questions regarding complexity of determining the hydra number of a graph which were left open in [1]. In particular we show that the problem of checking, whether a graph $G = (V, E)$ is single-headed, i.e. whether the hydra number of G is equal to the number of edges, is NP-complete. We also consider hydra number of trees and we describe a family of trees for which the hydra number can be determined in polynomial time.

1. Introduction

Hydra formulas were introduced in [1] as a special class of Horn formulas. A hydra formula is a definite Horn 3-CNF (i.e. conjunctive normal form where each clause consists of exactly three literals) φ satisfying that if $(x \wedge y \to z)$ is a clause in φ then so is $(x \wedge y \to u)$ for any other variable u (except x and y). A hydra is determined by the undirected graph G formed by the bodies in φ. Given a graph G we can also define its associated hydra function h_G which is defined by a hydra formula associated with G. Based on this we can define the hydra number $h(G)$ of G as the minimum number of clauses in a CNF representing h_G. Many properties of hydras were shown in [1,2]. It is easy to see that given a graph $G = (V, E)$ we have that $|E| \le h(G) \le 2|E|$. Graphs satisfying the lower bound are called *single-headed*.

In this paper we show that determining whether a graph is single-headed is an NP-complete problem. This answers an open question posed in [1]. This result is also an interesting addition to a long line of results concerning the Horn minimization problem, which is defined as follows: Given a Horn formula φ and a natural number k, determine whether there is an equivalent Horn formula ψ consisting of at most k clauses. The problem of determining the hydra number of a graph is a very special case of Horn minimization problem. The Horn minimization problem for definite Horn formulas was first addressed in [3] where its NP-hardness was established. Recently, it was shown in [4,5] that definite Horn minimization is not only hard to solve exactly but it is hard to approximate as well even when the input is restricted to definite Horn 3-CNFs. These results imply that definite Horn minimization is NP-hard already for 3-CNFs, a simpler proof of the same fact was recently provided in [6]. However in all the definite Horn 3-CNF related results the formulas produced by the respective polynomial reductions can have prime implicates of arbitrary size. Unlike that a prime implicate of a hydra formula φ always consists of exactly three literals. NP-completeness of determining the hydra number $h(G)$ of a general graph thus implies

E-mail address: kucerap@ktiml.mff.cuni.cz.

http://dx.doi.org/10.1016/j.tcs.2016.05.037

that the following restricted version of definite 3-Horn minimization is also NP-complete: Given a definite Horn 3-CNF φ which represents a Horn function whose all prime implicates are definite Horn clauses of size 3, and a natural number k, determine whether there is an equivalent Horn CNF ψ consisting of at most k clauses.

After considering the general case we turn our attention to the complexity of determining hydra numbers of trees. Some interesting results about hydra numbers of trees were already shown in [1]. It was shown in [1] that a tree $T = (V, E)$ is single-headed if and only if T is a star, and that $h(T) = |E| + 1$ if and only if T is a caterpillar. It was also shown in [1] that the hydra number of a complete binary tree $T = (V, E)$ is between $\frac{13}{12}|E|$ and $\lceil \frac{8}{7}|E| \rceil$. The complexity of determining the hydra number of a tree was left open in [1]. In this paper we make first steps in this direction by describing a subclass of trees such that for a tree T in this class it is possible to determine the value of $h(T)$ in polynomial time.

The paper is organized as follows. After giving necessary definitions and preliminaries in Section 2 we continue by showing the NP-completeness of determining the hydra number of general graphs in Section 3. In Section 4 we present a class of simple trees and a polynomial algorithm which determines their hydra number. In Section 5 we conclude the paper with some remarks and open problems. Due to the space limitations, most of the proofs are omitted.

A preliminary version of this paper was presented in [7].

2. Definitions and known results

In this section we shall introduce the necessary notions used throughout this paper.

2.1. Boolean functions

A *Boolean function* f on n propositional variables $x_1, \ldots, x_n$ is a mapping $\{0,1\}^n \to \{0,1\}$. The propositional variables $x_1, \ldots, x_n$ and their negations $\overline{x}_1, \ldots, \overline{x}_n$ are called *literals* (*positive* and *negative literals*, respectively). An elementary disjunction of literals is called a *clause*, if every propositional variable appears in it at most once. It is a well-known fact that every Boolean function f can be represented by a conjunction of clauses (see e.g. [8]). Such an expression is called a *conjunctive normal form* (or CNF) of the Boolean function f. A clause C is an implicate of a Boolean function f if C is satisfied on all assignments which are satisfying for f, it is a prime implicate if there is no proper subclause C' of C with this property. We shall say that two CNFs are *equivalent* if they represent the same function.

We shall often treat a CNF as a set of clauses. We say that a CNF φ representing a Boolean function f is *irredundant* if it is a set-minimal representation of f (i.e. for any clause $C \in \varphi$ we have that $\varphi \setminus \{C\}$ does not represent f). We say that a CNF φ is *prime* if it consists only of prime implicates (of the underlying function). The number of clauses in a CNF φ is denoted as $|\varphi|_c$.

A *definite Horn clause* is a clause in which exactly one literal is positive. We shall consider only the case of *definite Horn 3-clauses* which consist of three literals, one of which is positive and the other two negative, e.g. $(\overline{x} \vee \overline{y} \vee z)$, this is equivalent to implication $(x \wedge y \to z)$. The two variables appearing negatively in a definite Horn clause form its *body* and the only positive literal is called the *head* if this clause. E.g. in clause $(x \wedge y \to z)$, $\{x, y\}$ is a body of size two and z is the head. A *definite Horn (3-)CNF* is a CNF consisting of only definite Horn (3-)clauses and a *definite Horn function* is a Boolean function which can be represented by a definite Horn CNF.

In verifying that a given clause is an implicate of a given definite Horn function, a very useful and simple procedure is the following. Let φ be a definite Horn CNF of a definite Horn function h. We shall define a *forward chaining* procedure which associates to any subset Q of the propositional variables of h a set $FC_\varphi(Q)$ in the following way. The procedure takes as input the subset Q of propositional variables, initializes the set $FC_\varphi(Q) = Q$, and at each step it looks for a definite Horn clause $S \vee y$ in φ such that $S \subseteq FC_\varphi(Q)$, and $y \notin FC_\varphi(Q)$. If such a clause is found, the propositional variable y is included into $FC_\varphi(Q)$, and the search is repeated as many times as possible. The resulting set is called a *forward chaining closure of* Q *with respect to* φ (we omit φ when it is clear from the context). The following lemma, proved in [9], shows how the above procedure can help in determining whether a given clause is an implicate of a given CNF, or not.

Lemma 2.1. *Given a set $\mathcal{C}$ of pure Horn clauses, a subset Q of its propositional variables, and its variable $y \notin Q$, we have $y \in F_{\mathcal{C}}(Q)$ if and only if $Q \vee y$ is an implicate of the function represented by $\mathcal{C}$.*

2.2. Graphs

Throughout the paper we shall use standard graph notation (see e.g. [10]). A degree of a vertex v in a graph $G = (V, E)$ is the number of edges incident to v, a graph in which all vertices have degree 3 is called *cubic*.

Given graph $G = (V, E)$, the *line graph* $L(G)$ of G has vertex set $V(L(G)) = E$ and two edges $e, f \in E$ form an edge $\{e, f\} \in E(L(G))$ if they share a vertex, i.e. if $e \cap f \neq \emptyset$. A (vertex-disjoint) *path cover* of G is a set of vertex-disjoint paths such that every vertex $v \in V$ is in exactly one path. The *path cover number* of G is the smallest integer k such that G has a path cover containing k paths, the path cover number of G is denoted as $pc(G)$.

Given tree $T = (V, E)$, T^- denotes subtree of T formed by removing all leaves of T. A tree T is a *caterpillar* if T^- is a path. Equivalently T is a caterpillar if T^- does not contain a vertex of degree 3 or more. We say that a vertex $v \in V$ which